N92-22710

INCREASING PRODUCTIVITY THROUGH TOTAL REUSE MANAGEMENT (TRM)

M. P. Schuler
NASA Langley Research Center
Hampton, VA 23665

ABSTRACT

Total Reuse Management (TRM) is a new concept currently being promoted by the NASA Langley Software Engineering and Ada Lab (SEAL). It uses concepts similar to those promoted in Total Quality Management (TQM). Both technical and management personnel are continually encouraged to think in terms of reuse. Reuse is not something that is aimed for after a product is completed, but rather it is built into the product from inception through development. Lowering software development costs, reducing risk, and increasing code reliability are the more prominent goals of TRM. This paper describes procedures and methods used to adopt and apply TRM. Reuse is frequently thought of as only being applicable to code. However, reuse can apply to all products and all phases of the software life cycle. These products include management and quality assurance plans, designs, and testing procedures. Specific examples of successfully reused products will be given and future goals will be discussed.

WHAT IS TOTAL REUSE MANAGEMENT

Total Reuse Management (TRM) is achieved when an organization as a whole and its individual members make a commitment to continuously: search for reuse opportunities associated with each work product and activity; distribute notices to appropriate members of the organization of identified reuse opportunities; and actually reuse products, ideas, methods, and procedures. It is similar to Total Quality Management (TQM) in that its success depends on individuals adopting a mindset which influences all their activities, that is, reuse becomes a way of thinking and acting. With the expectation of increased productivity, management provides the necessary time and personnel for the record keeping and documentation required to support TRM reuse activities.

GOALS OF TOTAL REUSE MANAGEMENT

The major goal of TRM is to reduce software development costs by reusing previously developed knowledge and products instead of redeveloping them. Goals directly related to code development include reducing risk and enhancing reliability. The growth of reuse within a project decreases the number of elements to be developed which decreases the stress on the project schedule and thus lowers the risk of overruns. Since each time a component is reused it is retested in its new application, the additional error discovery and correction process naturally yields an enhanced product reliability. The greater the number of reused components the lower the overall risk of mission failure. Further, the successful demonstration in a previous project of a proposed method of implementation, increases confidence that the method is a worthy candidate for reuse. Another goal is process improvement since reuse can include procedures and methods as well as products. By documenting the improvements in procedures followed on a given project those improved techniques become available for reuse on future projects, thus improving the organization's procedures and methods over time.

THE TRM ADOPTION PROCESS

The first step is to instill a reuse mindset into the members of the organization. This was done from the top down at the Software Engineering and Ada Lab (SEAL). The SEAL adopted the idea of reuse as one of the lab's primary objectives. To achieve this objective, the regular weekly software management staff meetings were supplemented with discussions to promote reuse, determine individual project components which had reuse potential, and focus on reuse successes. Each project manager would report on products, concepts, and ideas which they considered candidates for reuse. In addition, project managers were required to report on products which had been successfully reused. Listing examples of these products gave other project managers ideas on where reuse might be possible on their project. Once the project managers began to adopt a reuse mindset the weekly reports were changed to monthly ones. Similarly, project managers were responsible for the introduction of TRM on their individual projects at project level staff meetings. In addition, a catalogue of various types of products reused was kept and is now being used to aid in training new employees so that they can quickly become contributors to the TRM program.

TYPES OF REUSE

The following sections give a variety of different areas and products in the software life cycle where reuse can be effectively established to lower development costs, reduce risk, and increase functional reliability. Reuse can occur across organizations, across projects, and within projects. These examples are given to illustrate the multiplicity of reuse situations which yield substantial benefits and to stimulate ideas concerning the applicability of reuse concepts at all organizational levels.

Reuse Through Technology Transfers Across Organizations

The Formal Inspection process is comprised of a set of procedures designed to efficiently detect and correct errors. Jet Propulsion Laboratory (JPL) had already developed a Formal Inspection Program and had years of practical experience with the process when the SEAL decided to adopt it as one of its measures to improve software product quality. JPL agreed to perform a technology transfer of information and products to the SEAL to support the initiation of the program. This saved the SEAL the hundreds of man-hours it would have cost to start up the program on their own. The SEAL was able to reuse JPL's managerial framework, training course materials, checklists, forms for conducting inspections, forms for recording inspection statistics, and a copy of their customized software to run the data base for statistical collection of inspection results. JPL also provided averaged cost estimates by software development phase for man-hour expenditures on inspections. By reusing their cost data the SEAL was able to calculate the estimated cost of inserting the inspection procedures into new projects. In addition, the two organizations agreed to communicate regularly to exchange information, documents, and data that could be reused. This is important since it assured that technological gains made at each site would be exchanged and reused.

JPL also trained SEAL staff members to teach their training course on formal inspections and those members are now actively involved in the process of conducting the same technology transfer to numerous organizations within NASA as well as different companies in industry. Not only are the formal inspection products and procedures being reused but the methods used to insert the process into the organization are also being documented and reused to further facilitate the transition. For example, for in-house projects, a set of procedures for introducing formal inspection has been established and is reused with each new project interested in inspections. For external projects, a notebook containing an insertion plan and all records of the execution of that plan is being compiled. All lessons learned from the insertion process and references to information used in the process, such as cost estimates, will be recorded in the notebook. Records tracking such things as number of inspections and actual cost will also be included. This information will be reused on future external projects.

Object Oriented Design (OOD) was another area of technology which the SEAL group was interested in. As the group recognized the need for incorporating OOD in its practices, the services of an organization expert in consulting and teaching OOD were employed. This substantially reduced the cost and time required to implement OOD technology within the SEAL. Once the insertion process was complete, an agreement was worked out between the expert organization and the SEAL group to exchange lessons learned and copies of completed designs for reuse on future projects. In addition the SEAL has developed an informal network to exchange information on OOD between several organizations and is now in the process of receiving reusable case study feedback from network members.

In both of these examples, Formal Inspections and OOD, the SEAL was successful at completely reusing technologies from outside organizations to improve its own software engineering capabilities. In addition, a core group of participants was established for each new technology to handle the future distribution of related reusable materials among the different organizations participating. This has resulted in an increase in productivity in two ways: by exchanging lessons learned and advancements in procedures followed by the individual organizations the core group has been able to achieve process improvements at a relatively rapid pace and low cost; and, by directly reusing completed products, considerable manpower has been saved. In addition the Seal makes it a point to regularly have informal technical interchange meetings with other organizations on specific areas of mutual interest. Such areas include: Systems Engineering, Object Oriented Design, the use of Ada programming, reuse, and risk

management as well as specific applications domain work. These meetings are conducted to transfer reusable technology between NASA sites and from NASA to Industry. The SEAL encourages and invites this type of interaction with all organizations interested in participating in technological transfers.[1]

Developing Expertise for Reuse within an Organization

Each individual in the SEAL organization was spending a considerable amount of time installing and becoming proficient in the use of new computer devices and software tools. A method was devised to reduce the overhead associated with these learning activities. For each new tool that was purchased, one or two individuals were given the responsibility of becoming experts on it. This involved searching through the technical reference manuals, learning how to set up the tools, and becoming proficient at the most common commands needed for typical use. Once this was completed, the experts reused their knowledge to install the tools on each computer and gave a brief seminar and list of common commands (along with explanations) to the general users to acquaint them with the tool in an expedient manner. This process saved the organization a considerable number of man-hours and was used on numerous devices and software tools such as: compilers, debuggers, logic analyzers, PC-based tools, PC-based networks, and specific bus protocols. If a tool was too complex to learn in a short seminar, the tool vendor was brought on-site to teach SEAL staff members how to give a detailed instruction course on the tools spectrum of functions and applications. Masters of the instructional materials needed to teach the courses were provided with the training. In depth classes on these tools are now given regularly with the only cost being the SEAL staff members salary and the cost of reproducing course materials.

The SEAL is also involved in bringing on-site a number of software engineering courses that are taught on a one time basis. A catalogue of course titles and attendants is kept so that if the need arises knowledge in that subject area can be easily tracked for reuse. Apprenticeships can also be set up to transfer knowledge for reuse within the organization. For example, one of the designers took on an apprentice at the beginning of a subsystem design and by the midway point the apprentice was a constructive contributor to the design. When the project was completed the apprentice took that newly gained expertise and reused it to design a subsystem on another project.

Finally, if training tapes were available from vendors they were purchased and reused throughout the organization. The cost of travel and tuition for technical training courses can put a substantial drain on an organization's budget. By reusing expertise, instructional materials and tapes, an organization can become educated in new software engineering technology at a much lower cost.

Reuse Within And Across Software Projects

The following is a description of how actual software products are being reused both within and across three different SEAL projects. All three projects are written in Ada and use Object Oriented Design Methodologies.

Project I[2]

Communication drivers are the software components developed to drive bus communication and they are among the most time consuming components to write. Designing them to be reusable would yield significant savings on future projects. This requires the use of a design approach which is both modular and hierarchial. A module is developed to serve as an interface between the applications level code and the code responsible for driving bus communication. The interface module is designed to provide a general purpose (application independent) interface to the bus while the associated bus modules are written to perform actual bus communications for a specific standardized bus protocol on a specific manufacturer's card. This promotes reuse in two ways. First, the applications level modules can be reused without modification if the bus should be replaced with one of a different type, since they had been developed with no dependencies on the type of communications hardware. Second, the software written to

1. For more information on technology transfers or training seminars contact Pat Schuler (804) 864-6732.
2. Copies of the code mentioned in this section are available courtesy of the Controls Structures Interaction Project. Contact Pat Schuler (804) 864-6732.

drive the bus communications can be reused on any project containing the same supporting communications hardware since the interface module was written to be applications independent.

Both of these types of reuse have been achieved on SEAL projects. For example, Project I used a MIL-STD 1553 bus in its PC-based Ground Support Equipment to test and record 1553 communications from a flight computer. The Ground Support Equipment, and over 4,000 lines of Ada code for 1553 communications, were reused to produce a similar checkout subsystem for a completely different flight project, Project II. In addition, the design for Project I was reused on a project in the same applications domain but with different hardware and software language requirements. This was possible since reusability had been a principle design requirement which assured that the completed product would not be dependent on the target hardware.

There were other types of communications software reuse within the same project. Project I contained three major subsystems and all three communicated across the 1553 bus. Subsystem I, used the 1553 bus to configure Subsystem II with information necessary to perform normal operations. Subsystem I was an INTEL machine and Subsystem II used a 1750A computer but the data structure used to hold the configuration data was identical on both systems. The data structure was developed on Subsystem II and reused on Subsystem I. Also, the data structures used to implement the remote terminal mode of the 1553 communications were reused without modification and the Ada package specification for both the bus control and remote terminal modes were reused between the subsystems with only slight modifications.

Another software component that was reused was a special purpose block move which was developed for Subsystem II. Its purpose was to increase throughput by transferring vectors rather than words of data from the 1750A to the array processor card and from the 1750A to the 1553 card contained within Subsystem II. This software was reused to improve communications rates within Subsystem III between its 1750A and digital signal processor card and also between its 1750A and 1553 card. Also, Subsystem III had an additional requirement. It not only had to accept configuration data from the 1553 bus but must be configurable via an RS232 bus. The code for reading in the user supplied information and processing it so that it could be used to set the configuration was reused from Subsystem I to fulfill this requirement for Subsystem III. This saved approximately 4 man-months of effort.

The ease with which the 1553, the block move, and the configuration code were reused is in a large part due to the use of Object Oriented Design (OOD) and Ada. OOD methods made it possible to minimize the communication between software modules while maximizing the cohesion between routines performing functions related to specific objects (or devices) in the system. The use of Ada constructs such as packages supported these methods and the portability of the language made reuse across different subsystems with different processors and bus interfaces possible. Further, each software component was required to use a standardized prologue stating specifics about the software it contained: compiler options, author's name, and date, etc. There were standards for how the software was to be written and formatted as well. Both of these contributed to readability which made software components easier to understand and therefore easier to reuse or modify for reuse.

Project I also reused, from another flight project, part of a math library written in assembly language. Although it had to be converted to 1750A assembly code the algorithms were reused. During the conversion process an error in one of the math routines was discovered and fixed on the 1750A code as well as the original source code. By reusing software across projects and subjecting it to different environments and applications, previously uncovered errors are found and fixed. The risk of future failure is decreased since its test domain has increased and that is a major advantage to reusing code. Its reliability increases the more it is reused. In the same way, when the 1553 code developed on Project I was reused for a different application on Project II, errors were discovered and the 1553 code was fixed on both projects and its reliability was increased as well.

System tests procedures can be reused as additional subsystem tests to eliminate errors prior to integration testing. This was accomplished by connecting Subsystem I and II with a simulator for Subsystem III which was not yet completed. These subsystems had already been individually tested, but performing the system integration tests on the hybrid system acted as an additional suite of tests and uncovered errors which would not have been discovered until system integration testing. Further, since the hybrid system had now been verified, when errors did occur during full system integration testing, the errors were contained to either the newly added subsystem or communications

with that system. This substantially reduced the time spent in debugging. Therefore, reusing system integration test procedures as additional unit tests can flush out problems early and help isolate problems as each new subsystem is added.

Project II[3]

Project II reused the standardized prologue developed on Project I which insured consistency in software component documentation across projects. The Ground Support Equipment and the Mil-STD 1553 bus code were also reused from Project I. The 1553 bus communications drivers were reused without modification because they had been developed with a general purpose interface module as the delimiter between the applications specific software and the hardware specific software. This ability to reuse not only the code but the actual computer equipment provided Project II with a substantial savings.

Project II achieved even better results with code reuse between its subsystems. The project set up a directory where completed code was placed so that it was available for all project members to reuse. Among the products stored in this directory are software components to perform: windowing capabilities; text and menu displays; and a general purpose parser for converting transmission packets. In addition it also contains software for; UART / serial communications via the RS232 protocol, queues, ring buffers, semaphores, data structures, and time packages which provide information such as the julian date and system time. Each of these components are being reused within Project II across several subsystems and this has substantially reduced the total cost of the system being developed. Project II has produced a substantial amount of code for INTEL hardware which is available for reuse. A complete Ada interface for all registers and modes has been written for the following; INTEL 8259 Programmable Interrupt Controller, INTEL 8254 Programmable Interval Timer, INTEL 8237 Programmable DMA Controller, INTEL 8255 Programmable Peripheral Interface, and INTEL 8274 Multi-Protocol Serial Controller. In addition, a complete Ada interface for all internal registers of the INTEL 80186 microprocessor has been written and is available for reuse.

Project III

This is the third project started since the SEAL began the TRM program. It will be reusing the 1553 bus components and the coding prologues that were developed on Project I as well as the coding standards from Project II. In addition, a substantial amount of code will be reused from Project II such as software to perform; windowing capabilities, text and menu displays, keyboard input manipulation and filtration, and RS232 protocol. It will also reuse packages which provide functions to; manipulate linked lists, implement the semaphore construct, declare varying length strings, log events with time stamps, and replay prerecorded user input for batch testing. It will also reuse many of the software components developed for INTEL chips mentioned above.

It is clear from the three project examples that a substantial amount of reusable code can be accumulated in a relatively short amount of time. However, it is important to recognize that the successful reuse of code on these projects was largely due to developers regularly reporting on the completion of potentially reusable software components. Even more important was the staff's commitment to build each component to be reusable. To facilitate this, modern design techniques such as those found in Object Oriented Design were used along with the Ada programming language which provides constructs to support those design methods.

Software Development Documentation Procedures That Encourage Reuse

Software documentation is a major expense on any software system. The key to reducing that cost is to use methods and procedures that result in development documents which can then be reused for formal reviews and for the required deliverable documentation. This can be done at each phase of the software lifecycle. During requirements analysis phase event sequences and responses, data elements, timing order and constraints as well as states and processes can all be documented in tabular, graphical, or textual form as part of the stepwise analysis

---

3. Copies of code mentioned in this section are available courtesy of the Lidar In-Space Technology Experiment. Contact Chuck Carpenter (804) 864-8046.

process. These documents are the output products of the requirements phase. If careful attention is paid to thoroughly documenting each step of the process those documents can be reused, without alterations, in the requirements document and for the requirements review. Once the requirements phase is complete, object diagrams with written object descriptions and operation descriptions as well as a requirements traceability matrix and decomposition tree showing the hierarchical connections in the design can be used to thoroughly document the activities required to complete the preliminary design phase. These output documents, again without modification, are reused as the deliverables for the preliminary design and the preliminary design review. If all of these documents are routinely updated as part of the activities for the detailed design and coding phases, they may also be reused not only for the detailed design review and test readiness review but could be reused for the 'As Built Configuration Document' which describes the final product. In addition, automated testing procedures along with confirmed test results can be reused in the users/operators manual as diagnostics. The 'As Built Configuration Document' and the test procedures can also be reused during maintenance phase. They are invaluable to the maintainers as a record of the system's purpose and how the functional specifications were achieved. As modifications are completed these documents are updated so that they can be reused for the life of the maintenance phase.

In addition to the above-mentioned reuse benefits within a project, these documentation techniques encourage reuse across projects. Since the products are well documented and the documents are kept current, other projects will find it easier to understand the product's requirements and functions. This ease of understanding makes the chore of locating reusable components less time-consuming and therefore more attractive. In addition, since not only the code but the requirements and design are available for each component, savings from reuse can also be achieved during requirements and design phases on new projects.

For this type of document reuse to be achieved a defined process with specific steps and procedures must be in place before a project starts. These procedures need to specify the form and content of all output products for each step and their function as input documents into the next phase. Instead of the developers producing large quantities of documentation that they incorrectly perceive as the customer's needs, a true picture of the current project posture can be obtained from reusing the actual development documents as the review documents.

Reusing Documents From Other Organizations And Projects

When the reuse program started, the SEAL had no documentation of its own to reuse. Therefore, an effort was made to obtain existing documents from numerous organizations within government and industry. A documents library was started and currently holds project documents covering almost every phase of the software lifecycle: Software Requirements, Interface Requirements, Test Procedures, Management Plans, Configuration Control Plans and Verification and Validation Plans. Portions of these are reused each time a new document for a software project is written and this has saved a considerable amount of man-hours. Procedures for constructing testing trees or a traceability metrics, for example, need only be written once and can then be reused in the documentation for other software projects. The library also holds documents on how to perform specific software engineering tasks such as quality assurance, design, benchmarking compilers and a variety of more general documents on software engineering standards and procedures. All of these are reused on SEAL projects to determine various software development procedures. In addition, several contracting organizations responsible for developing software for the SEAL have also reused many of these in developing deliverable software documents and establishing their own software procedures.

Standardizing On An Environment Promotes Reuse

The SEAL has a standard environment for software development which includes; computers, compilers, debuggers, logic analyzers, etc. The SEAL has also standardized on word processors, drawing tools, spreadsheet programs and data base managers. To make reuse easier, each staff member is connected to a local area network. Network licensees were purchased for the tools in the standardized environment. That proved to be more economical than purchasing individual copies. Status reports and presentations can be constructed by reusing information electronically obtained from the various nodes on the network. The tools are chosen based on their compatibility to support such activities. Forms are constructed using these standardized tools, for documenting common activities

such as purchase requests and travel requests and inventorying books, software and hardware, etc. Those forms are made available to all users of the network to reuse.

FUTURE DIRECTIONS

Once documentation is completed, all the SEAL software components will be submitted to COSMIC (the NASA reuse library) for general access. Project data such as man-hours, lines of code and pages of documentation being collected on current projects will be reused. The data will be entered into existing off-the-shelf estimating tools such as PRICE and REVIC. Entering actual data from SEAL projects into tool data bases will customize the output to the SEAL environment and potentially give more accurate project cost estimates. Even more extensive metrics will be recorded on future projects and will be reused to perfect further project estimates. In addition, descriptions of completed code written for specialized hardware used on SEAL projects will be given back to the vendors of those hardware products. They will be encouraged to relay this information to their customers via newsletters or computerized bulletin boards so that organizations purchasing those hardware products may have the opportunity to reuse completed SEAL code. The SEAL also has plans to develop its own local library of reusable components to hold software for such things as ring buffers, stacks, math library routines, etc. Software components for specific chips and busses will also be catalogued and entered in the library. The SEAL is currently promoting standardization on specific busses and chip sets. Reuse can be maximized by using standardized hardware and reusing existing custom-built software and test environments to substantially cut future project costs. A software project handbook will also be written to provide information on established methods and procedures for reuse on future projects. In addition, work is currently being done to develop a set of generic management, quality assurance, testing, etc., plans that will act as templets to be reused by all future projects and project specific information will be inserted to customize the plans. To promote reuse on a center-wide scale, a Software Engineering Users Group is being started which will meet regularly to exchange information on software methods, procedures, tools, and completed modules.

CONCLUSION

If the current activity will ever be performed again, there is potential for reuse. Perform the activity accordingly so that it or its products can be easily reused and document its existence so others can locate, understand, and reuse it.